Curso de Introdução a Python
Aula 05: Estruturas de repetição

Ana Luiza Martins Karl

Sumário

01

Introdução

- Estruturas de repetição

Introdução

Executam um bloco de código repetidamente através de condições especificas

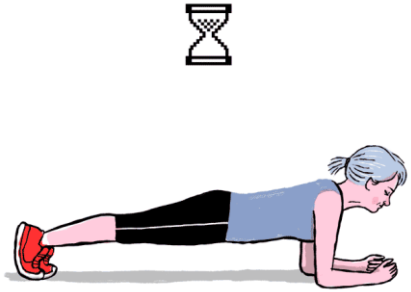
- **Automatização de tarefas repetitivas**
- **Redução de código redundante**
- **Análise de grandes volumes de dados**

Introdução

Executam um bloco de código repetidamente através de condições especificas

Laço *for*

Laço *while*



02

Laço *for*

Laço *for*

O *for* é utilizado para percorrer ou **iterar** sobre uma **sequência de dados** -- seja uma lista, uma tupla, uma string -- executando um conjunto de instruções em cada item.

Sua sintaxe básica é: for <nome variável> in <iterável>

```
for item in iterable:
    # código a ser executado
```

Laço *for*

O *for* é utilizado para percorrer ou **iterar** sobre uma **sequência de dados** -- seja uma lista, uma tupla, uma *string* -- executando um conjunto de instruções em cada item.

```
dna_sequence = "ATGCTTCAGAAAGGTCTTACG"
nucleotide_count = {'A': 0, 'T': 0, 'C': 0, 'G': 0}
for nucleotide in dna_sequence:
    if nucleotide in nucleotide_count:
        nucleotide_count[nucleotide] += 1
print(nucleotide_count)
```

Laço *for*

O ***for*** pode ser usado com o ***else***, de maneira que os comandos do bloco ***else*** serão executados quando as condições do ***for*** não forem satisfeitas

```
for item in sequencia:
    print(item)
else:
    print('Todos os items foram exibidos com sucesso')
```

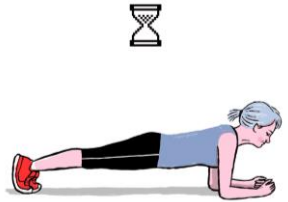
03

Laço *while*

Laço *while*

O **while** é uma estrutura de repetição utilizada quando queremos que determinado bloco de código seja executado **enquanto** determinada condição for satisfeita.

```
while condição:
    # código a ser executado
```



Laço *while*

O ***while*** é uma estrutura de repetição utilizada quando queremos que determinado bloco de código seja executado **enquanto** determinada condição for satisfeita.

```
count = 0
while count < 5:
    print(count)
    count += 1
```

Laço *while*

Assim como o **for**, o **while** também pode ser utilizado junto com a condição **else.** O bloco de comandos do **else** será executado quando as condições do **while** não forem satisfeitas

```
contador = 0

while contador < 10:
    contador += 1
    print(f'Valor do contador é {contador}')
else:
    print(f'Fim do while e o valor do contador é {contador}')
```

04

Auxiliadores de controle de fluxo

Controle de fluxo

Um conceito importante:

Controle de fluxo refere-se à maneira como a sequência de instruções é executada em um programa de computador.

Lembrem-se das aulas anteriores: o **Python é uma linguagem interpretada**, então há execução linha-a-linha.

Logo, as **estruturas de decisão e repetição**, além de outros comandos são **estruturas de controle de fluxo** pois são capazes de alterar a ordem de execução de um programa.

Controle de fluxo

- Determina a ordem na qual as instruções, declarações ou funções são executadas ou avaliadas.
- Permite a criação de algoritmos complexos.
- Facilita a tomada de decisões e repetição de tarefas.
- Torna o código mais flexível e eficiente.
- Essencial para lidar com diferentes cenários e dados de entrada.

Controle de fluxo: *pass*

O **pass** é um operador que não realiza operação nenhuma.
É um marcador para um código futuro.

```
if condição:
    pass  # ainda não foi implementado
```

É utilizado para estruturas de decisão e repetição com o bloco vazio.

Controle de fluxo: *continue*

O **continue** pula a iteração atual e continua com a próxima iteração do laço.

```
for nucleotide in dna_sequence:
    if nucleotide == "G":
        continue
    print(nucleotide)
```

Controle de fluxo: break

O **break** Interrompe a execução do laço imediatamente

```
for nucleotide in dna_sequence:
    if nucleotide == "G":
        break
    print(nucleotide)
```

Obrigada!